A new matrix multiplication systolic array

Patrice Quinton, Brigitte Joinnault, Pierrick Gachet

INRIA

CENTRE DE RENNES

IRISA

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tel (1) 39 63 55 11

Rapports de Recherche

N° 525

A NEW MATRIX MULTIPLICATION SYSTOLIC ARRAY

Patrice QUINTON
Brigitte JOINNAULT
Pierrick GACHET

Mai 1986

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTÈMES ALÉATOIRES

Campus Universitaire de Beaulieu
35042 - RENNES CÉDEX
FRANCE
Téléphone : 99 36 20 00
Télex : UNIRISA 950 473 F
Télécopie : 99 38 38 32

Publication Interne n°290 – Avril 1986
12 pages

A NEW MATRIX MULTIPLICATION SYSTOLIC ARRAY[1]

International Workshop on Parallel Algorithms and Architectures,
Luminy, Marseille, France, 14 – 18 April, 1986

Patrice QUINTON
Brigitte JOINNAULT
Pierrick GACHET

IRISA, Campus de Beaulieu,
35042 RENNES – Cedex
FRANCE

Résumé : On présente un nouvel algorithme systolique pour le produit de matrices. Il a la propriété que les résultats peuvent être réutilisés dès leur production pour la multiplication suivante sans être mémorisés. On montre comment cette architecture peut être utilisée pour le calcul de polynômes matriciels ou pour les puissances d'une matrice. La construction formelle de cet algorithme à partir des équations du problème est donnée.

Abstract : A new systolic array for matrix multiplication is presented. It has the property that the results of one multiplication can reenter the array and be used in the next multiplication without memorizing intermediate results. We show how this array can be used for the calculation of a matrix polynomial, and for matrix powers. The formal derivation of this systolic array is described.

---

[1] This work was partially supported by the French Coordinated Research Program C3.

1 Introduction

Matrix multiplication is one of the earliest algorithm for which a systolic implementation scheme was found (Kung and Leiserson, 1978). It received considerable attention (Hwang and Cheng, 1982), (Melkemi and Tchuente, 1985) because of its practical importance in signal processing as well as in numerical analysis. To our knowledge, only one systolic array for matrix multiplication allows the results to be used immediately for the next step. Such a design will be said to be **result reusable**. It was described by (Culik, 1982) as an example of the use of folding techniques in order to obtain systolic algorithms. On the other hand, Muroga (1984) presents a systolic array for the calculation of the product of three $n \times n$ matrices, but his design needs a memory for intermediate results.

In this paper, we describe a suprisingly simple result reusable systolic array for matrix multiplication. This algorithm was discovered by looking at the dependency mapping procedure of successive matrix multiplication steps of the form $X^{(n)} = X^{(n-1)}A$ where A, $X^{(n)}$, and $X^{(n-1)}$ are matrices. This formal synthesis will be described in section 3; however, we feel that a simple informal explanation given in section 2 will help understanding this design. In section 4, applications of this new design to the matrix polynomial calculation and to the computation of matrix powers are presented.

2 Informal presentation of the result reusable matrix – multiplication systolic array

It is well – known that the matrix multiplication, $C = AB$, can be implemented on two dimensional arrays. Numerous designs can be used depending on whether one wants C, A or B to move. Here we consider the simple design where C stays in place. Coefficient c_{ij}, where $1 \leq i,j \leq n$ thus is calculated by cell i,j of a mesh connected array of multiply – and – add processors, as depicted by figure 1. Cell i,j contains a register c which is first reset to zero and then accumulates successively products $a_{ik}b_{kj}$ where k varies from 1 to n. Note that this accumulation can be done with efficiency 1, i.e. the processors are effectively doing one calculation per systolic cycle. However this implementation suffers from the drawback that the results do not move and consequently a systolic output scheme has to be designed in order to recover them.

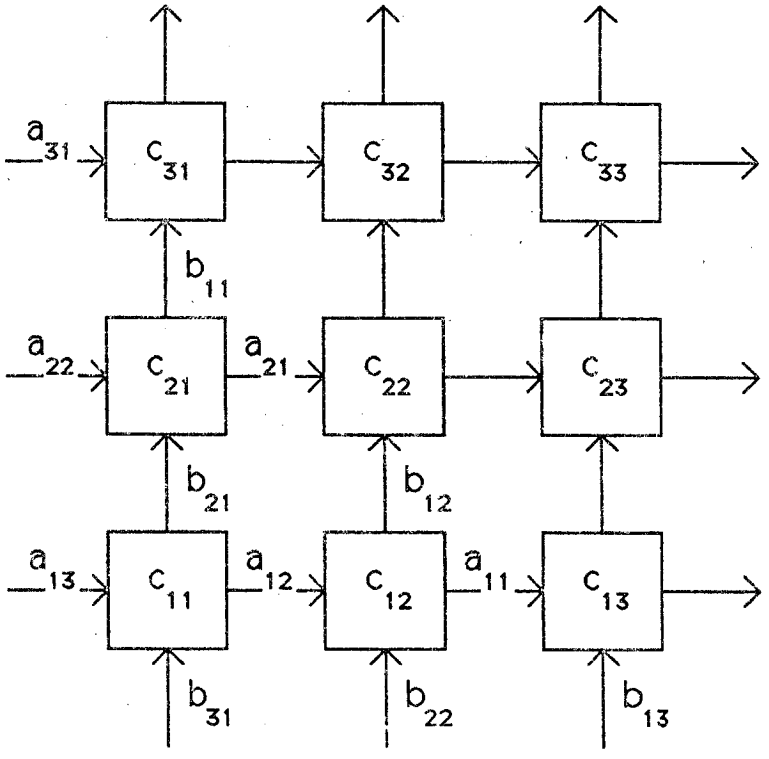

Figure 1. Systolic array for the multiplication of matrices

Suppose that one wants the results c_{ij} to be reused for the next multiplication step, i.e., they should enter the left column of the array. Consider row i of the array of figure 2 and assume that a new right to left link is provided between the cells. The final value c_{ij} is obtained when the last element a_{in} visits cell i,j. When this occurs, we can then send the final result to the left, and j steps later, the result reaches the left cell of the array. However, the rate at which the c_{ij}'s leave the array is only half the rate at which the a_{ij}'s enter it. This phenomenon is similar to the Doppler effect in physics. Consider the time t at which a_{in} reaches cell i,j. Then, at time $t+1$, a_{in} reaches cell $i,j+1$. But c_{ij+1}, which is sent to the left neighboring cell needs one more step to reach cell i,j. Therefore, cell i,j sees the successive coefficients, c_{ij}, c_{ij+1}, ..., c_{in} respectively at time t, $t+2,\ldots,t+2(n-j)$.

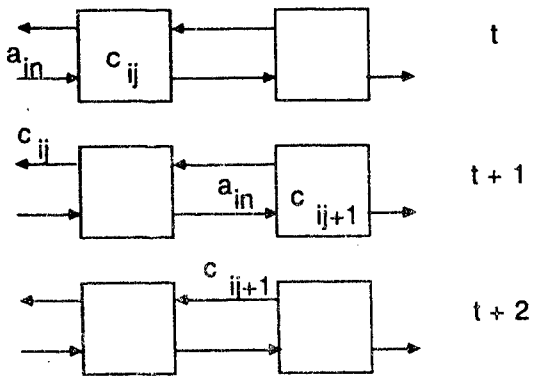

Figure 2. Illustration of the Doppler effect in systolic arrays

A very simple solution to this problem consists in slowing down by a factor two the input rate of *A* and *B*, without any other change. Figure 3 shows the design that is obtained. Each cell of the left column is provided with a feed back loop controlled with a multiplexer which makes it possible to select either the input of a new matrix *A*, or to feed back the result of the previous multiplication.

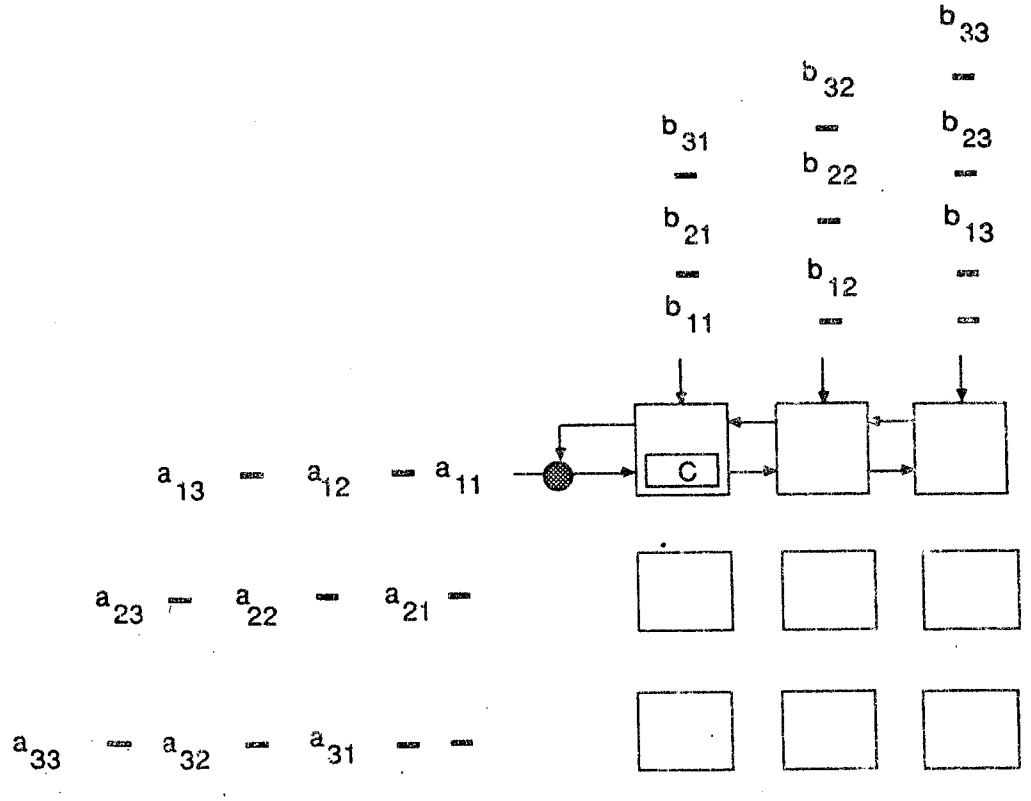

Figure 3. The result reusable systolic array for matrix multiplication

3 Formal derivation

We now describe how this design can be derived from the equations of the algorithm, following the dependency mapping synthesis procedure. Consider the problem of multiplying two square matrices $X = AB$, and chaining this operation with $Y = XB$. In this section, we shall follow the synthesis method described by Quinton (1983 ; 1984) which consists in describing the calculations using uniform recurrence equations as defined by Karp et al. (1967), then schedule and map these calculations by means of linear timing and allocation functions. We start with

$$x_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \tag{1}$$

A *left-to-right serialization* of the Σ operator gives

$$\begin{aligned} &X(i,j,k) = X(i,j,k-1) + a_{ik} b_{jk} \\ &X(i,j,0) = 0 \\ &x_{ij} \equiv X(i,j,n) \end{aligned} \tag{2}$$

As a_{ik} is common to all equations having i and k fixed, and similarly, b_{kj} is common to all equations having k and j fixed, we can replace the first equation of (2) by

$$\begin{aligned} &X(i,j,k) = X(i,j,k-1) + A(i,j,k).B(i,j,k) \\ &A(i,j,k) = A(i,j-1,k) \;\; \textit{if } j>0 \;\; \textit{else } A(i,0,k) = a_{ik} \\ &B(i,j,k) = B(i-1,j,k) \;\; \textit{if } i>0 \;\; \textit{else } B(0,j,k) = b_{kj} \end{aligned} \tag{3}$$

The domain of computation associated with (3) is a cube $D = \{(i,j,k) \mid 1 \leq i,j \leq n\}$ depicted by figure 4. Let $t(i,j,k)$ denote the time at which computation at point (i,j,k) can be done. Then it can be easily checked out that $t(i,j,k) = i + j + k - 3$ is consistent with the partial order imposed by the dependencies between the calculations (see Quinton, 1984 for more details). Systolic arrays can be obtained by a regular allocation of the calculations to processors (using for example linear allocation functions such as projections). In our case, a projection of the cube along axis k gives exactly the design described by figure 1.

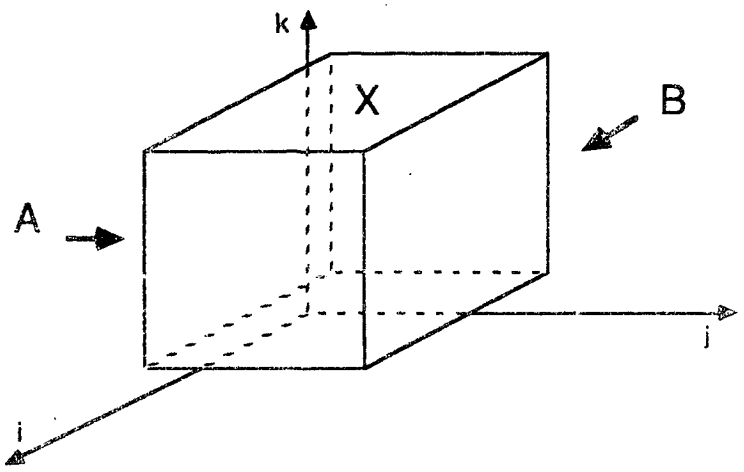

Figure 4. Domain of computation associated with the matrix multiplication

In order to chain the matrix calculations $X = AB$ and $Y = XB$ on the same design, one has to superpose the cube D_1 of figure 4 and another cube D_2 representing the new computation $Y = XB$, as represented on figure 5. The same transformations that were applied to obtain equation (3) now produce

$$\begin{aligned} Y(i,j,k) &= Y(i,j,k-1) + X(i,j,k).B(i,j,k) \quad \textit{if } k > n \textit{ else } 0 \\ X(i,j,k) &= X(i,j-1,k) \quad \textit{if } j > 0 \textit{ else } x_{ik} \\ B(i,j,k) &= B(i-1,j,k) \quad \textit{if } i > 0 \textit{ else } b_{kj} \end{aligned} \tag{4}$$

where $n + 1 \leqq k \leqq 2n$. The difference between (3) and (4) lies in the fact that x_{ik} in (4) is the result of calculation of (3). But x_{ik} is produced at point (i,k,n) of D_1, and has to be routed to point $(i,0,k+n)$ of D_2. This can be done by k elementary movements along the vector $(0,1,-1)$.

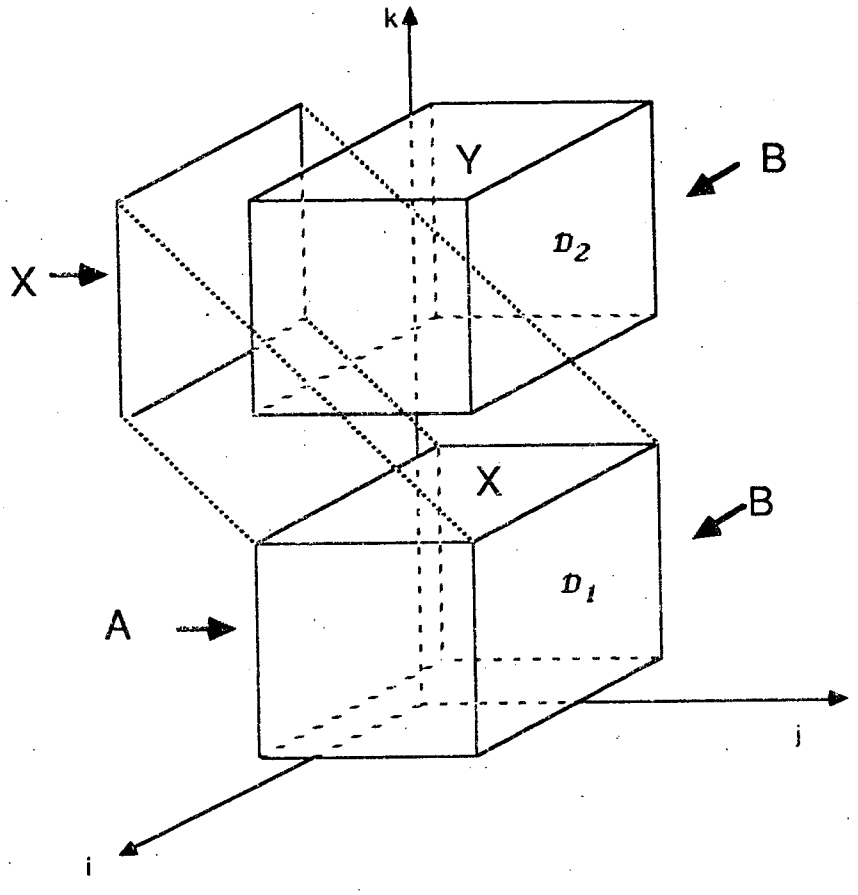

Figure 5. Domain of computation associated with the chaining of two matrix multiplications

Therefore, equation (4) can be rewritten as

$$
\begin{aligned}
&Y(i,j,k) = Y(i,j,k-1) + X_2(i,j,k).B(i,j,k) \;\; \textit{if } k > n \;\; \textit{else } 0\\
&X_2(i,j,k) = X_2(i,j-1,k) \;\; \textit{if } j > 0 \;\; \textit{else } X_1(i,j,k)\\
&X_1(i,j,k) = X_1(i,j+1,k-1) \;\; \textit{if } k > n \;\; \textit{else } X(i,j,k)\\
&B(i,j,k) = B(i-1,j,k) \;\; \textit{if } i > 0 \;\; \textit{else } b_{kj}
\end{aligned}
\qquad (5)
$$

By routing x_{ik}, we have introduced a new dependence vector $(0, 1, -1)$. The timing-function $t(i,j,k) = i + j + 2k - 3$ is compatible with this new dependency. By projecting the domains D_1 and D_2 along axis k, we obtain exactly the design of figure 3.

4 Applications of the systolic array

In this section, we describe two different applications of the result reusable systolic array. The first one concerns the calculation of matrix polynomials, and the second one matric powers.

4.1 Matrix polynomial

A slight modification of our design allows us to perform matrix multiply – and – add step of the form $X^{(n)} \leftarrow X^{(n-1)}A + B$. The accumulation of B can be done on the fly when the values $X^{(n-1)}A$ reach the left row of the array, as depicted by figure 6. Consider the calculation of the matrix polynomial $P = \Sigma_{k=0}^{N} B_k A^k$ where B_k and A are $n \times n$ matrices (the algorithm is also valid when the matrices are $n \times p$, or when the coefficients B_k are vectors). Using Horner's rule, P can be computed by the following iterative scheme :

$X^{(0)} = B_N$
$X^{(k)} = X^{(k-1)}A + B_{N-k}$
$P \equiv X^{(N)}$

Therefore, P can be computed in exactly $2n(N + 1) - 1$ steps on the systolic array of figure 6.

4.2 Powers of a matrix

A well – known efficient algorithm for the computation of matrix powers $P = A^N$ has been described by Knuth (1969, pp. 398 – 422), and Lamagna (1982). It consists in using the binary representation of N to control a sequence of square or multiply by A steps. Although this method is not optimal, it has the greatest advantage, compared to others, that it does not require an important storage. This property makes it well – suited for a systolic realisation.

More precisely, let $N^p N^{p-1} \ldots N^0$ be the binary representation of N, and let $C_q, C_{q-1}, \ldots, C_0$ be the new sequence obtained by rewriting each N_k as SX if $N_k = 1$ or as S if $N_k = 0$, and by discarding the first pair of letters SX. C is used to control a sequence of operations and is interpreted from left to right, each S meaning « square the current matrix» and each X « multiply the current matrix by A ».

For example, $19 = 10011$ will be rewritten as $SSSXSX$ and the sequence of computations will be $A^2, A^4, A^8, A^9, A^{18}, A^{19}$.

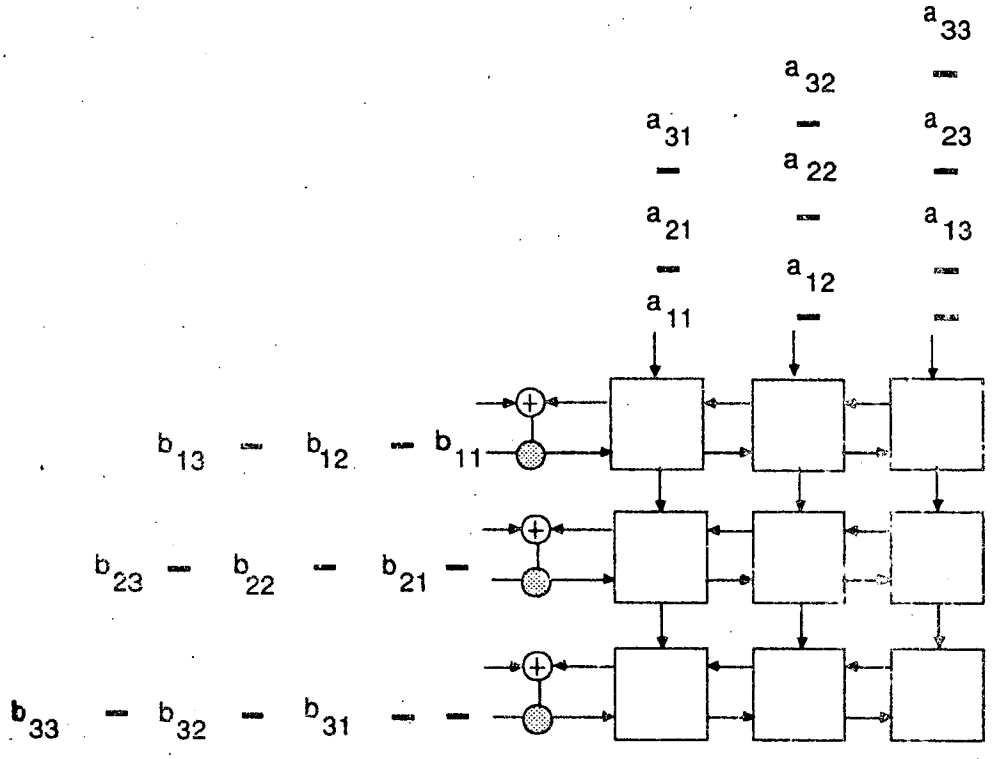

Figure 6. Systolic array for the matrix polynomial calculation

Then the following iteration scheme

$X^{(0)} = A$
$X^{(n)} = (X^{(n-1)})^2$ *if* $C_{q-r} = S$ *else* $X^{(n-1)}A$
$P \equiv X^{(q)}$

provides A^N. The basic calculation to be done is either a square or a matrix multiplication.

Again, a simple modification of our design enables us to square a matrix. The basic idea is to route the result also to the upper row of the array. Depending on the value of C_{q-r}, A or $X^{(n-1)}$ will be fed into the array from the top to the bottom.

If we consider again the diagram of figure 5,this operation corresponds to moving the results of the upper face of D_1 to the foreground face of D_2. This can be done by elementary movements along the vector (1,0, – 1). This new dependence vector is compatible with the timing – function $t(i,j,k) = i + j + k - 3$ and the resulting design is shown by figure 7.

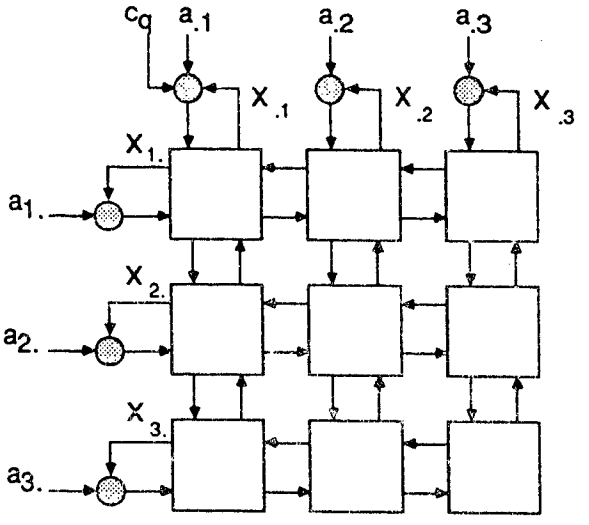

Figure 7. Systolic array for the powers of matrix calculation

As the binary representation of N has $\lfloor \log_2 N \rfloor + 1$ bits, q is majored by $2\lfloor \log_2 N \rfloor$. Hence, the calculation of A^N takes a maximum of $2n(2Log_2 N + 1) - 1$ steps on a n^2 mesh connected array.

5 Conclusion

A new very simple systolic array for the multiplication of matrices has been presented. It has the main advantage that the results can be reused efficiently for a new multiplication as soon as they are produced, without the need of intermediate storage. We have shown that this design can be used for the calculation of matrix polynomials in time $O(nN)$, and for the calculation of A^N in time $O(n\lfloor LogN \rfloor)$. We have shown how these designs can be obtained formally using geometric transformations of the equations of the algorithms. The work presented in this paper is a contribution to an effort aiming at defining formal techniques enabling systolic architectures to be designed automatically from high-level specifications. We believe that this effort is essential for the definition of fast and reliable CAD tools for systolic chips design.

6 References

(Culik and Pachl, 1982)
K. Culik and J. Pachl, "Folding and Unrolling Systolic Arrays," ACM SIGACT – SIGOPS Symp. on Principles of Distributed Computing, Ottawa (1982).

(Hwang and Cheng, 1982)
K. Hwang and Y – H. Cheng, "Partitioned Matrix Algorithms for VLSI Arithmetic Systems," *IEEE trans. on Computers*, Vol. C – 31, No. 12, Dec. 1982, pp. 1215 – 1224.

(Karp, 1967) R.M. Karp, R.E. Miller, S. Winograd, "The Organization of Computations for Uniform Recurrence Equations," JACM, Vol. 14, N0. 3, juillet 1967, pp. 563 – 590.

(Knuth, 1969) D. Knuth, "The Art of Computer Programming," Vol. 2, Seminumerical Algorithms, Addison Wesley, 1979.

(Kung and Leiserson, 1985)
H.T. Kung and C.E. Leiserson, "Systolic Arrays (for VLSI)," in J. Duff and G. Stewart, eds., *Sparse Matrix Proceedings* (SIAM, Philadelphia, 1978) pp. 256 – 282.

(Lamagna, 1982) E.A. Lamagna, "Fast Computer Algebra," *Computer*, Sept. 1982, pp. 43 – .

(Melkemi and Tchuente, 1985)
L. Melkemi and M. Tchuente, "Programmation du produit matriciel sur un reseau systolique rectangulaire," *TSI*, Vol. 4, No. 5, 1985, pp. 459 – 469.

(Muroga, 1984) C. Muroga, "On a case of Symbiosis between Systolic Arrays," *INTEGRATION, the VLSI journal* 2, (1984), 243 – 253

(Quinton, 1983) P. Quinton, "The systematic Design of Systolic Arrays," IRISA Research Report, No 193, March 1983.

(Quinton, 1984) P. Quinton, "Automatic Synthesis of Systolic Arrays from Recurrent Uniform Equations," 11th Annual Int. Symp. Computer Arch., June 1984, Ann Arbor, pp. 208 – 214.

PI 282 **Stabilité robuste dans la commande adaptative indirecte passive**
Philippe de Larminat – 70 pages ; Janvier 86.

PI 283 **Data synchronized pipeline architecture pipelining in multiprocessor environments**
Yvon Jégou, André Seznec – 36 pages ; Janvier 86.

PI 284 **An overview of the GOTHIC Distributed Operating System**
Jean-Pierre Banatre, Michel Banatre, Florimond Ployette – 24 pages ; Janvier 86.

PI 285 **Optimal sensor location for detecting changes in dynamical behavior**
Michèle Basseville, Albert Benveniste, Georges Moustakides, Anne Rougée – 32 pages ; Février 86.

PI 286 **La tolérance aux fautes dans un système temps-réel à contraintes strictes**
Maryline Silly – 32 pages ; Février 86.

PI 287 **A new statistical approach for the automatic segmentation of continuous speech signals**
Régine André-Obrecht – 38 pages ; Mars 86.

PI 288 **Synthèse sur les réseaux locaux temps-réel**
Philippe Beimans – 40 pages ; Mars 86.

PI 289 **Calcul distribué d'un extrémum et du routage associé dans un réseau quelconque**
Jean-Michel Hélary, Aomar Maddi, Michel Raynal – 36 pages ; Mars 86.

PI 290 **A new matrix multiplication systolic array**
Patrice Quinton, Brigitte Joinnault, Pierrick Gachet – 12 pages ; Avril 86.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique